BACHELOR'S THESIS

Degree in Mathematics

Title: Using heterogeneity for obtaining the Medial Axis Transform of natural images

Author: Gemma Canet Tarrés

Advisor: Sven Dickinson, José Tomás Lázaro (tutor)

Department: Computer Science

Academic year: 2017-2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Abstract

From Blum's Medial Axis Transform, that obtains the skeleton of binary images by joining the centers of all maximal inscribed disks in the shape, the AMAT was developed. The Appearance-MAT is a method that aims to compute the medial axes for natural images, as well as an appearance-based reconstruction. However, when considering the appearance, it deals only with color, and requires image smoothing to obtain a better result. As an approach to solving this problem, while abandoning the goal of reconstruction, we present and evaluate ten different methods for obtaining the medial axis transform, all based on the concept of heterogeneity. The idea behind this ensemble of methods is that in any natural image, any maximal inscribed disk should be homogeneous both in color and texture, while the disks that are not completely contained in an object should exhibit a much higher heterogeneity.

Keywords: Medial axis transform, shape skeleton, figure-ground segmentation, computer vision

Contents

1 Historical background **3**

2 Introduction **5**

3 Related work **6**

4 AMAT **7**

5 Heterogeneity as a substitute for the reconstruction error in AMAT **9**
5.1 Color and luminance histograms 13
5.2 Texture histogram 13
5.3 Histogram distances 14
5.4 Computing heterogeneity score 15
5.4.1 Method 1 16
5.4.2 Method 2 17
5.4.3 Method 3 19
5.4.4 Method 4 20
5.4.5 Method 5 21
5.4.6 Method 6 21
5.4.7 Method 7 22
5.4.8 Method 8 23
5.4.9 Method 9 23
5.4.10 Method 10 24

6 Testing the methods **24**
6.1 Preprocessing 24
6.2 Evaluation of the model 26

7 Results **27**

8 Conclusions and Future Work **35**

1 Historical background

When a human sees an image, it's automatically able to distinguish every element in it, to tell what's in the background and what's in the foreground, and to explain how everything relates to each other and what kind of scene the image depicts. However, for a computer, an image is just a matrix where every pixel is represented by three numbers: their intensity of red, green and blue color, each being a value between 0 and 255. Therefore, getting it to distinguish the different elements that are in it, whether they are in the foreground or background, can be extremely challenging for researchers working in computer vision. It's for this reason that abstraction, grouping, segmentation and classification problems are so important in this field and have been attempted to be solved in many different ways throughout the years.

When certain features are recognized in an image, in order for them to be understandable and abstracted, a certain processing needs to be done. If the abstraction is parts-based, all non-accidental relations between the features can be useful for grouping them. These relations can appear in different forms in the image. They can be appearance-based, such as color and texture affinity, or shape-based, such as contours or regions.

This last affinity has been studied by many researchers throughout the years, who have tried to solve the shape-based object recognition problem in many different ways. However, although the grouping of causally related features is necessary for object categorization, it has been proven not to be sufficient, creating the need for studying a more abstract level of shape features. For this reason, the abstraction of shape has evolved and has been studied in many different ways. Let's see a few of the most important milestones in this field [1, 2, 3].

First of all, in the 1970's, Binford presented an approach based in generalized cylinders [8, 9]. The idea behind this was to model every 3D shape as a deformed cylinder, defined by its axis, represented by a space curve, and a set of cross-section and sweep functions that specify the size and shape transformation of the cross-section along the axis. As all these elements could get very complicated, some restrictions such as a straight axis, a homogeneous sweep function, a linear sweep function or a rotationally symmetric cross-section had to be introduced.

These restricted generalized cylinders offered a good way of categorizing abstract object models (Figure 1A), but presented many difficulties in the recovery of different parts and relations from images of real objects. This method was definitely not perfect, but it inspired more researchers to pursue the idea of modeling 3D using shapes with a symmetry axis. One of the most well-known methods that came from this era was the use of superquadric ellipsoids [10], which are generalized cylinders with a wide range of deformations with just a small set of parameters. This method usually gave good results for abstracting models from a 3D shape (Figure 1B), but struggled to deal with recovery of 2D images, for the same reasons that the restricted generalized cylinders failed.

Another approach created by Biederman in the same era was the use of geons [11]. Geons are nothing else than simple shapes such as cylinders or boxes that

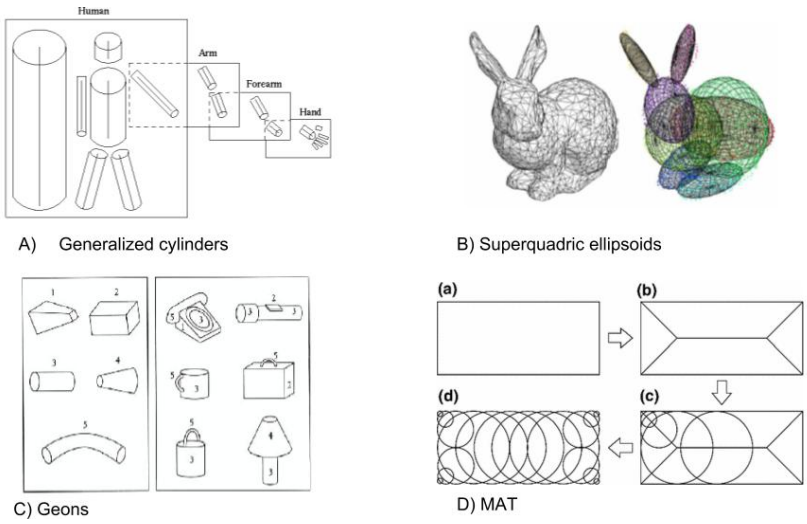

Figure 1: A) Human body built from generalized cylinders. Different levels of abstraction are shown in the arm. (Image obtained from [4]). B) Left: Mesh representing a rabbit shape. Right: Ellipsoidal representation with smooth boundaries and 12 ellipsoids. (Image from [5]). C) Left: Sampling of geons. Right: Common objects composed of geons. (From [6]) D) For stages of the MAT: a) Binary image of the object, b) Medial axis of the object, c) Sample of maximal disks inside the object, d) MAT with a sample of the disks. (From [7])

can be easily recognized by humans and represent the basic units for making up an object (Figure 1C), agreeing with the principles in Gestalt psychology [12]. These principles claim that all objects and scenes can be identified by abstract shape. In this case, it would state we can perceive each geon individually and group them into objects. However, this method eventually had to face the same challenges as the previous ones and could not find a solution either, because of the unrealistic assumption they were all making that all features comprising the models could be directly observed in the image, driving researchers to pursue new ideas.

Trying to follow a different approach, Blum [13, 14] presented the medial axis transform (MAT), a 2D axial symmetry-based shape description of the image which assumes that all points in an object's contour map one to one with the output points of the model. The method operates by identifying a set of maximal disks inside an object that cover all the pixels, and obtaining its axis by joining the resulting centers of these disks (Figure 1D). This method, however, was introduced in the context of binary shapes, where all contour points are easily identified. But, since it doesn't accommodate either color or texture, and when working with natural, cluttered images, the contour points can easily be occluded or unclear, then the need for some previous image abstraction appears.

This method has been very useful for researchers over the years, since it's

been proven to provide a compact representation of the original shape, used for achieving a higher efficiency in algorithms dealing with shapes in many different ways [15, 16, 17]. For this reason, it's easy to understand how its limitations when applied to natural images have driven many researchers to attack the problem and try to find a plausible solution.

2 Introduction

Our work is presented as an attempt to achieve a generalization of Blum's Medial Axis Transform for natural images. The starting point, however, is not the MAT itself, but the AMAT, the Appearance-MAT presented by Stavros Tsogkas and Sven Dickinson in 2017.

This method, based on the MAT and developed in section 4 of this report, associates each medial point with a scale and an encoding, leading to two important improvements. The first one is inspired by the invertibility property of the MAT and is the reconstruction of the image, while the second one is the decomposition of the different regions that define the medial axis, by clustering all medial points into different branches.

This method is evaluated on the Berkeley Medial AXes (BMAX500), a dataset of medial axes based on the BSDS500 database, showing a better reconstruction quality with respect to three different baselines and with 90% compression. However, the encoding associated to each of the medial points used for both abstraction of the axis and reconstruction of the image is only based on color, not texture. For this reason, the method does not perform well when the image is very textured, so a certain smoothing of all input images needs to be performed as a preprocessing step for the AMAT extraction.

In this project, we intend to present an approach to solving this problem, by ignoring the reconstruction and just focusing on the position and scale of the medial axis points for a textured natural image. Getting back to the binary MAT, we know the medial axis points should be the centers for maximal disks inside of shapes. So, an idea on how to extend this definition into natural images could lie in the concept of heterogeneity.

All disks completely contained in a shape with regular appearance should be quite homogeneous, both in color and texture, while disks that present some pixels inside the shape and some pixels outside of it should be more heterogeneous (Figure 2). Therefore, with this idea in mind, we could define the medial points of a natural image as the centers of the maximal inscribed homogeneous disks, with this homogeneity referring not only to color, but also to texture and lightness.

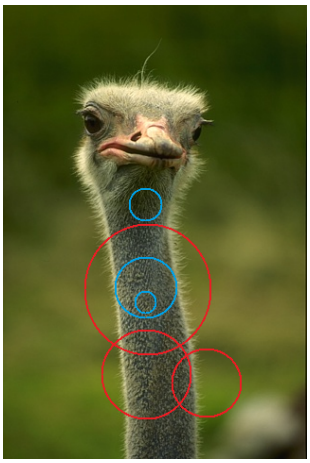

Figure 2: Examples of possible disks computed on a natural image from dataset BSDS500. Blue disks are homogeneous and red disks are much more heterogeneous.

The way of computing heterogeneity in a disk, however, is not straightforward, so in this project we'll introduce different methods of doing it and compare the results we get when performing each one of them on the BMAX500 dataset against the ones obtained by the AMAT.

3 Related work

Inspired by Blum's MAT, many researchers have been trying to follow his footsteps, exploring different extensions, modifications and applications of his work. Recently, some have tried to model the MAT as a classification problem and decide whether pixels are medial points or not by using machine learning algorithms. Some examples of this approach are the use of CNNs with side outputs of Shen et al [18], or the Multiple Instance Learning approach proposed by Tsogkas and Kokkinos [19].

In a more traditional approach, from the idea of representing shapes by using a skeleton, shock graphs were first introduced by Siddiqi et al. [20]. While shocks are defined as the singularities of a curve evolution process acting on a shape's boundaries, shock graphs are directed, acyclic graphs that represent the distribution and organization of such singularities.

The shock graph introduced a partitioning of the medial axis transform into branches based on qualitative shape.

To address the instability that both skeletons and shock graphs introduced, the bone graph was created [21]. This new representation focused on the different parts of a silhouette and its relations, capturing the ligature structures of the medial axis and leading to a more stable response with better performance in view-based object recognition.

With the same goal of abstracting shape, but working on natural images, the idea of multi-scale blobs arose [22]. This approach identifies parts of the objects by placing elongated or round symmetric structures on them, and studying the relationships between them by focusing on the edges of the structures. Although providing great regularization and obtaining good results when used for shape recognition, they still fail when dealing with textured images.

It's interesting to notice that most of the methods that intend to abstract shape from an image have a strong connection to symmetry. This is not a coincidence, since symmetry, and specially local symmetry, is exhibited in many objects in the world and is frequently used by humans for identifying and analyzing what they see. It's for this reason, and also because of its invariance to viewpoint, that it plays such an important role in perceptual grouping of image structure.

Levinstein et al. [23] drew on this idea in a method for detecting and grouping symmetric parts of objects in natural images, by creating deformable disks, considered as superpixels, and merging them to define the branches of the skeleton that's obtained from its medial points. With the same goal of obtaining the medial axis points in natural images, but in a different approach, the AMAT is presented and aims to obtain the medial axis transform by formulating and solving a weighted geometric set cover problem.

4 AMAT

This method, developed by Stavros Tsogkas and Sven Dickinson [24], has the goal of obtaining the skeleton and reconstruction of natural images (Figure 3). For doing so, it follows different steps.

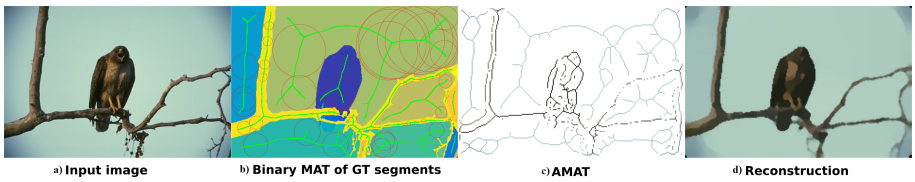

Figure 3: Example of the steps of the AMAT. (a): Input image I from dataset BSDS500. (b): Segmentation with color-coded ground-truth segments, including medial axes (green) and a subset of medial disks $D_{p,r}$ (red). (c): skeleton resulting from the AMAT. (d): Reconstructed image from the AMAT in c). (Image extracted from [24])

In the first step, given an image I (such as Figure 3a), it encodes disks $D_{p,r}$ of different radius r centered at each pixel of the image, which we will refer to as p. The encoding $f_{p,r}$ of each disk $D_{p,r}$ is a vector of three numbers corresponding to the average of red, green and blue that denote all the pixels of the original image contained in the disk. From that encoding, for obtaining the reconstructed image, a decoding function g can be defined and assigned to

every pair of pixel p and radius r as a function $g_{p,r}$ that maps $f_{p,r}$ back to a disk patch $\tilde{D}_{p,r} = g_{p,r} = g \circ f_{p,r}$. Using these two functions, the reconstruction error can be defined at each pixel p as the difference between the disk of radius r centered in p in the original image and the reconstructed one:

$$e_{p,r} = \|\tilde{D}_{p,r} - D_{p,r}\|$$

So, by trying to minimize the sum of the reconstruction errors while picking enough disks to cover the whole image, the AMAT is defined as the set of tuples $M : (p_1, r_{p1}, f_{p1,r_{p1}}), ..., (p_m, r_{pm}, f_{pm,r_{pm}})$, such that:

$$M = \arg\min_{p,r} \sum_{i=1}^{m} e_{p_i,r_i}$$

$$I = \bigcup_{i=1}^{m} D_{p_i,r_i}$$

and the set of points $p_1, p_2, ..., p_m$ are grouped into branches, building the skeleton of the image.

The disks that fulfill the previous conditions and therefore take part in the medial axis transform (such as the ones in Figure 3b) are chosen by defining a geometric set cover problem and solving it by using a greedy approximation algorithm. The weighted geometric set cover problem is formulated as follows. Consider a finite set of radius $\mathcal{R} : r_1, r_2, ..., r_R$ and a set $\mathcal{D}$ of r-disks with $r \in \mathcal{R}$. Taking the spatial support X^I of an input image I as the universe of N points $p = (x, y)$, all r-disks can be placed at any of these points such that $D_{p,r} \subset X^I$. Each (p_i, r_j)-disk gets assigned a cost c_{ij}:

$$c_{ij} = \sum_{k} \sum_{l} \|f_{ij} - f_{kl}\|^2, \qquad \forall k, l : D_{kl} \subset D_{ij}, i \in [1, N], j \in [1, R]$$

which receives a low value if the encoding f_{ij} is similar to the encoding of all disks fully contained in D_{ij}, meaning that p_i could be a good candidate for being a medial point. The goal, as stated before, is to find a subset of disks that cover the whole image, while maintaining a low total reconstruction cost.

It should be noted that all costs are calculated in the CIELAB color space, meaning that the pixel RBG values have been transformed into values L, A and B, corresponding to luminance, color component green-red, and color component blue-yellow, respectively. With our data presented this way, non linear relations between the different channels are similar to the nonlinear responses in our eyes, so that uniform changes of components in this space correspond to uniform changes in perceived color, making it a more suitable color space for measuring perceptual distances [25].

Getting back to the algorithm, once all disks have been assigned with a cost, it should start finding the disks that define the MAT. For doing that, since it's looking for a sparse solution for efficiency, we would like to choose the maximal

disks. However, this is not an easy thing when working with natural images, since boundaries are not always clearly defined, and that's why the AMAT adds a scale-dependent term $s_j = \frac{w_s}{r_j}$ to the costs c_{ij}, so that the selection of larger disks is favored. The value w_s used in the previous definition corresponds to a certain fixed weight that's been defined experimentally, and which has been proven to obtain a good balance between reconstruction quality and sparsity of the results.

Once this WGSC is defined, it's a strongly NP-hard problem that can be solved with existing polynomial-time approximate solutions. The chosen greedy solution used in this case computes the costs c_{ij} for all disks D_{ij} and defines the effective cost of each disk as

$$c_{ij}^e = \frac{c_{ij}}{A_{ij}} + s_j,$$

where A_{ij} is the number of pixels covered by D_{ij} and that haven't been covered by any other disk. Once this is defined, we start from an empty set M and add the disk with the lowest effective cost to the solution, while removing the area covered by D_{ij} from the remaining pixels to be covered and recomputing the costs of all disks that intersect with it. Then, the process is repeated until all pixels are covered.

In conclusion, the AMAT provides an efficient, useful way of adding appearance to the Medial Axis Transform and achieves good results when computing the skeletons of natural images. However, although it works really well on smooth images, the results are not as good when the images have textures. When applying the encoding and decoding to a textured image, the encoding computes the average of the colors contained in each disk, without dealing with the texture, so the resulting disk patches are smoothed. As a consequence, the decoding function gives a completely smoothed image as the reconstructed one, and the reconstruction error is too high in the parts where the texture was very significant.

The solution the AMAT proposes for this issue is based on smoothing the whole image as a pre-processing step. By doing this, the input image is already a smooth one and, as we've seen, the algorithm gets good results on these ones. However, in this thesis, we suggest a different way of solving it by relaxing the appearance side of the algorithm and giving up on the reconstruction part to focus on the ability of getting the medial axis points on textured images without the need of any kind of pre-processing.

5 Heterogeneity as a substitute for the reconstruction error in AMAT

The idea of relaxing the appearance side of the AMAT is the starting point of the method we are presenting. We want to find a way of detecting the maximal inscribed disks in every object of the image we are working with, and we want it to work on textured images. As seen in the section before, we want to pick

the disks in the image that are inscribed in a certain object or figure, i.e., the ones where color, light and texture are regular along the whole disk. It's this exact thought that gives us the idea of using the concept of heterogeneity. Heterogeneity is defined as the quality or state of being diverse in character or content, and homogeneity, its opposite, is therefore the quality or state of being all the same or all the same kind. Therefore, what we are trying to find are the maximal homogeneous disks that cover the whole image.

Translating this into the weighted geometric set cover problem presented in the AMAT, where the goal was to find the set of disks that cover all the pixels while achieving a low cost, what we want in this case is to find the subset of disks that cover the whole image while minimizing the global heterogeneity score. For doing so, we have substituted the cost

$$c_{ij} = \sum_k \sum_l \|f_{ij} - f_{kl}\|^2, \qquad \forall k, l : D_{kl} \subset D_{ij}$$

assigned to each disk D_{ij} in AMAT with a heterogeneity score.

For calculating heterogeneity, we first have to translate the image from RGB space to CIELAB color space, for an easier computation of perceptual distances, and also translate it into black and white for dealing with texture. After these translations are completed, the calculation of heterogeneity score for each disk can be performed in many alternative ways, since it's not an already stablished concept.

When we have an image where all parts have a very similar color and the texture is quite regular throughout the whole image, it's safe to assume the picture is very homogeneous and therefore should have a low heterogeneity score. If we add a certain irregularity to that image, let's say a stain of a different color or an object of different texture, then the heterogeneity should grow with the size and degree of difference of this irregularity. It's going to be higher if the object has different color and texture than if it just had different color but the same texture, or if texture was the same but it presented a different color (Figure 4).

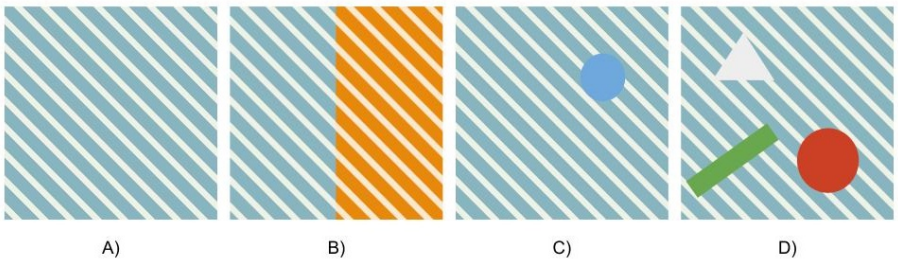

Figure 4: Sample of images with different homogeneities. A) The entire image has the same regular pattern, so it's very homogeneous. B) Texture is the same along the image, but color changes, increasing its heterogeneity. C) There's not a big variation in color, but texture is not the same along the image, making the image less homogeneous. D) There are very different colors and many texture variations, making the image very heterogeneous.

Although it can seem very simple, images can be extremely complicated, and computing these scores is not always simple. Some images can show patterns that are better understood by computing the scores in a certain way, while others can lead to problems if we apply the same method. For example, we could divide the disk in parts and compare each of them to the whole patch, but the number of subpatches we should compare is prohibitely expensive.

We could divide the disk into four parts, in 16 or even in more parts. However, as these parts keep getting smaller, they are also getting closer to the most basic level. Let's see this with an example. Consider a checkerboard image such as the one illustrated in Figure 5.

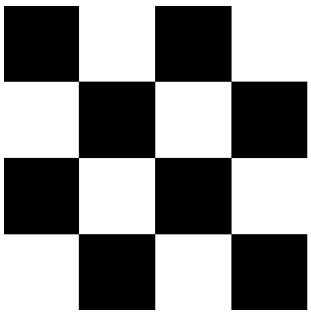

Figure 5: Checkerboard image, representing regular patterns that can lead to confusions when computing their heterogeneity scores

When a human looks at the picture, they see homogeneity, since it's all covered by a regular pattern. If we divided the 4x4 patch into four 2x2 parts, and compare each of them to the whole image, we would obtain high homogeneity, since the proportion of black and white pixels doesn't change, and we are

comparing color and texture, but not geometrical distribution, as we will see. Therefore, we would obtain the expected answer.

If we go one more level down, dividing the whole image in 16 parts, all subpatches will be either completely white or completely black, reaching the most basic level, where nothing is going to change if we go even deeper and divide the image into more parts. When this happens, by comparing each of the 16 resulting subpatches to the whole image, we get a much higher heterogeneity than we did before, yielding a result different than the one we would like to obtain. And here we see the importance of considering the right number of subpatches for computing the heterogeneity scores of every patch in the image.

For efficiency reasons, we won't consider dividing patches into more than 16 parts, so we are going to consider dividing each disk into 4 subpatches and into 16, as shown in Figure 6. In order to compare them more easily, we are going to name parent patch to the whole disk, children subpatches to the division in the left and grandchildren to the 16 subpatches in the right.

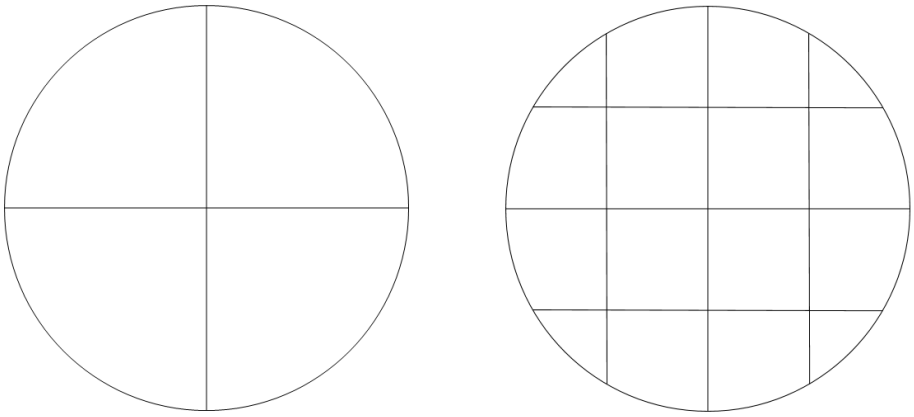

Figure 6: Left: Disk patch divided into 4 children subpatches. Right: Disk patch divided into 16 grandchildren subpatches

By choosing one decomposition or the other, and combining the results in different, plausible ways, many methods for computing heterogeneity scores can be defined.

To compare each of the selected subpatches to the parent patch or against each other and obtain a heterogeneity score, we first need to find a way of identifying the relevant information for performing such comparison. The features we'll be comparing for deciding the heterogeneity of each patch are colors A and B, luminance, and texture, since they are the ones that define its appearance. And for expressing them in a compact format that's easy to compare, we'll compute the histogram for each of these features in the patch we are studying.

5.1 Color and luminance histograms

Let I be the image defined in the CIELAB space that we are working with. I is a matrix of size H x W x C, where H is the image's height, W its width and C is 3, corresponding to the channels that characterize the CIELAB space: color A, color B and luminance. Since a histogram is a representation for the distribution of certain values into different bins, we first need to define the number B of bins in which we want to distribute our data. We use a default value of $B = 32$. Then, all values in each channel of I should be assigned to a bin $b \in [1, B]$ in order to create the distribution.

This distribution is what's going to make it possible to compare the different parts in every disk and evaluate its degree of heterogeneity. For more efficiency in the code, we compute the histograms that define all parts in the image at the beginning, and for identifying each of these parts, we define a set of filters $\mathcal{F}$. Each filter F in $\mathcal{F}$ is a binary matrix of size H x W that defines a certain subpatch. For example, if we are considering a distribution that includes the parent patch and its 4 children subpatches, we will have a set of 5 filters for each scale, one of them being a disk and the other ones being each of its four parts.

This way, considering one channel, c, in I at a time, for all values $b \in [1, B]$, we consider the binary matrix I_b representing all values in channel c of I assigned to the bin b. By convolving I_b with a filter F, we get all b values inside of the patch described by F, and therefore, by convolving all I_b matrices with the whole set of filters, we define the histograms of the disk patches centered at all points in the image, as well as the ones describing its different children and grandchildren subpatches.

5.2 Texture histogram

The main contribution of the approach we are presenting should be the ability to deal with textures, and it's for this reason that we should also encode the texture histogram for all patches and subpatches in the image.

For doing so, we use the black and white version of our image, since texture is independent of color. We first want to bin it into B different values, for the same reason we did in the previous case, but we now do it using a texton approach [26]. For using this approach, we first need some set of filter responses, which are clustered by k-means. Each of these clusters defines a texton, which is nothing else than a linear combination of different filters. Finally, by assigning every pixel in the image to the nearest texton, the combination of filter responses that's more similar to the patch, we get a histogram distribution in what we call a texton approach. It should be noted that, since we want a fixed number of bins B for the resulting histogram, we'll need to have B textons so that, when assigning each pixel to one of them, we obtain the binned image that we are looking for. Then, after I is binned, we can compute the texture histogram for every patch in the image the same way we did for the other channels.

This histogram gives an idea of the texture shown in each part of the image

and is used as a tool for differenciating each of the shapes that appear in a natural image. Two disks will only have similar texture histograms if they belong to the same object or have a very similar appearance.

5.3 Histogram distances

As we'll see in the next section, for computing the different heterogeneity scores, we will have to compare different regions of the image multiple times. For example, if dividing a parent patch into different parts, in any of the structures described in Figure 6, we may want to compare each of the children subpatches to the whole disk, the parent patch. For doing so, we should first of all consider the regions X_1 and X_2 of the image I such that $X_i \in I$, $i = 1, 2$ that we want to compare.

Let H_1 be the set of histograms for lightness, colors A, B, and texture that define the region X_1, and H_2 the set that describes X_2. We will refer to each histogram in the set H_i as h_{ij}, where $i \in 1, 2$ denotes the region and $j \in 1, 2, 3, 4$ is for the channel described by the histogram. For comparing the regions X_1 and X_2, we should compare sets H_1 and H_2, and the way we do this is by going one by one through all the channels in the regions, computing $distance(h_{1j}, h_{2j}), \forall j \in 1, .., 4$ (Figure 7).

For computing these distances, we use the χ^2 distance, since it's useful for comparing discrete probability distributions and it's symmetric on both distributions. We define it as:

$$distance(h_{1j}, h_{2j}) = \frac{(h_{1j} - h_{2j})^2}{(h_{1j} + h_{2j})}$$

and apply it to each of the 4 values of j, obtaining a set of four distances d_j, representing the differences between all the features in the two regions we are comparing.

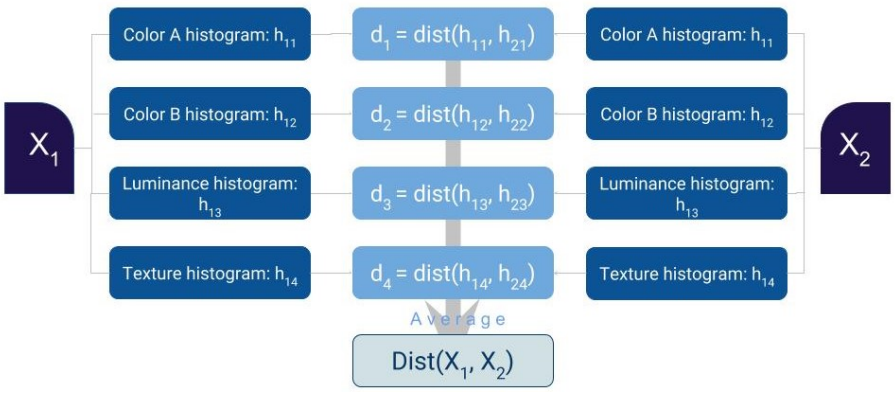

Figure 7: Diagram representing the steps for calculating the distance between two regions X_1 and X_2

Since we want to have a single value that defines the similarity between X_1 and X_2, we need a way of combining the 4 values d_1, d_2, d_3, d_4 into a single one. We could just take the maximum d_j value, and associate the highest difference between features as the value between the patches, but we may be losing information, since we would be assigning the same difference to patches with just one very different feature and to the ones where all features are very distant, so we should try to use another method, like a weighted sum.

One could argue that lightness is more important than color for defining similarity, or that two patches cannot be related if their texture is very different, but there are certainly occasions in which one feature takes precedence over the other ones and other situations in which another feature is dominant. Ideally, we could learn the weights that give better results on a set of images and assume those are the ones that will perform better overall. However, for the sake of simplicity and for keeping this approach as an unsupervised method, we take the average of the four values as the distance between the two regions, getting:

$$distance(X_1, X_2) = \frac{1}{4} \sum_{j=1:4} \frac{(h_{1j} - h_{2j})^2}{(h_{1j} + h_{2j})}$$

5.4 Computing heterogeneity score

Once we know how to compute the different gradient-based features in any region of interest in the image and how to compute their differences, we finally need a way of combining them to obtain the degree of heterogeneity that defines the region. For doing so, we can use any of the two divisions shown in Figure 6 and compare all subpatches in different ways. Let's now explore a number of ways we can obtain these scores, all represented in Table 1.

	p vs c	p vs g	c vs g	c vs c	combine c	combine
Method 1	compute	compute				average
Method 2	compute		average		average	average
Method 3	max	max				max
Method 4	compute		average		average	max
Method 5	compute		average		max	max
Method 6	max		max			max
Method 7	average					
Method 8	max					
Method 9				average		
Method 10				max		

Table 1: Summary of all methods studied in this report. First column represents how the method combines the distances between the parent patch and its children subpatches, the second column represents how it does with the distances between parent and grandchildren, the third column does the same with distances between children and grandchildren, and in the forth column how distances between all of the children subpatches are combined. Finally, in the fifth column it's shown how all distances involving the children subpatches are combined, and in the last column, how all the measurements are combined to obtain the final heterogeneity score. Blank spaces mean that nothing is combined.

5.4.1 Method 1

When looking at the AMAT algorithm, we see the cost assigned to a certain disk D_{ij} is obtained by averaging the difference between the encodings of all the smaller disks completely contained in it and the encoding of the disk itself. Translating that into our method, the most straightforward thing is to assume we should compare the histograms of every disk D_{ij} with the histograms of all its possible subpatches, and average all the resulting distances. Since comparing the whole disk to absolutely all possible patches contained in it would take too long to compute and would slow down the algorithm, the closest thing to do by using the grids presented before in Figure 6, is computing both 4 coarser parts obtained from the first grid and the 16 finer ones from the second grid.

If we name the histograms that define each of the children parts as s_{1i}, with $i \in 1, ..., 4$, the histograms from the 16 grandchildren subpatches as s_{2j}, where $j \in 1, 2, ..., 16$, and the ones obtained from the parent patch as p, we should compute

$$D_{1i} = dist(p, s_{1i})$$

for all values of i,

$$D_{2j} = dist(p, s_{2j})$$

for all values of j, and average them, obtaining the patch has a heterogeneity score of

$$H = \frac{1}{20}\left(\sum_{i=1}^{4} D_{1i} + \sum_{j=1}^{16} D_{2j}\right)$$

The more subpatches are found to be different to the parent patch, the higher H will be. Therefore, the method can be very useful for assigning a certain heterogeneity score to patches with a significant irregularity and making this score proportional to the size of the irregularity. However, in some cases, this method can lead to some unwanted results. This is the case of the previous checkerboard example (Figure 5), where the comparison of all smaller subpatches to the whole disk contributes to make H higher than it should be. This could maybe be fixed by not just averaging all children and grandchildren subpatches, but assigning each of the two groups a weight that's proportional to their area, giving more importance to differences in the coarser parts than in the finer ones.

Another problem of this method is its lack of rotational invariance, meaning that the scores obtained for two patches with the same pixel distribution but with a certain angle offset between them, such as the ones presented in Figure 8, will have a different value.

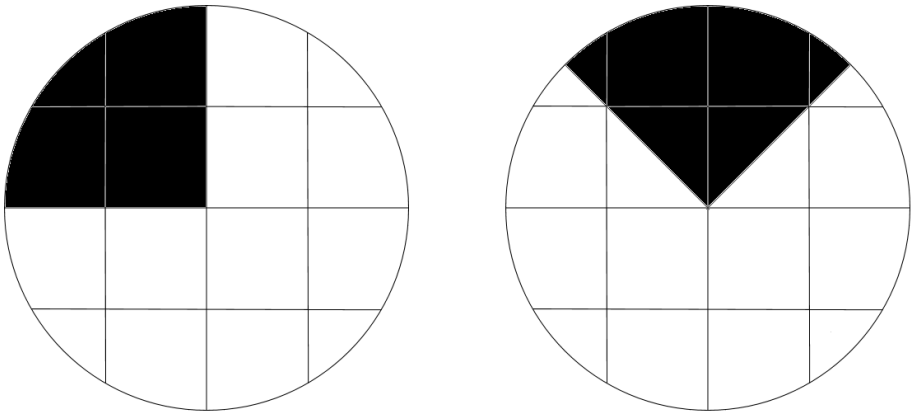

Figure 8: Disks with different heterogeneity scores in method 1

The parent patch in both cases shows both black and white pixels, although with a higher proportion of white ones. All subpatches in the first disk, on the other hand, are either all black or all white, while the second disk presents also some subpatches with half black pixels and half white ones, both in the finer subpatches and the coarser ones. This creates a larger number of subpatches that differ more from the parent patch in the left image than in the right one, making heterogeneity values get higher in the first case. For trying to achieve more equal heterogeneity scores in cases like this, a second method is developed.

5.4.2 Method 2

Rotational invariance with the use of squared grids is not easy to obtain. However, with the right comparisons between all subpatches, where we don't compare the finer subpatches to the coarsest ones, but to the immediate coarser ones, the method can become way closer to invariance. In this algorithm, we

pretend to find the average of the distance between the distributions in the whole disk and the ones in each of its 4 children subpatches, by comparing also each subpatch to its own 4 children parts. For computing the final score, we should first define each of the 4 children subpatches of the disk as S_i, where $i \in 1, ..., 4$. By dividing S_i into 4 parts, the values of s_{ij} get defined, with $j \in 1, ..., 4$, representing the histograms that describe each subpart in S_i, each grandchildren of the disk (Figure 9).

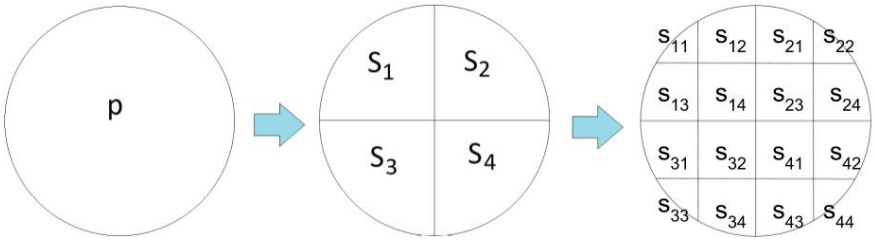

Figure 9: Definition of patch and subpatches in method 2, 4, 5 and 6

With all parts defined, we compare each grandchildren patch to the children one where they belong, and each children subpatch to the parent patch p:

$$d_{ij} = dist(S_i, s_{ij})$$

$$D_i = dist(S_i, p)$$

Then, the heterogeneity h_i inside each coarser subpatch S_i can be defined by the average of distances between it and its 4 children parts s_{ij}:

$$h_i = \frac{1}{4} \sum_{j=1}^{4} d_{ij}$$

Finally, by averaging the distance of each subpatch S_i to the whole disk with its heterogeneity h_i, we get the value that describes the quadrant:

$$q_i = \frac{1}{2}(h_i + D_i)$$

And the heterogeneity score is obtained as the average of the values describing each part of the disk:

$$H = \frac{1}{4}\sum_{i=1}^{4} q_i$$

With this method, the heterogeneity score assigned to the different patches is more rotational invariant than in the method before, but it's also true that when bringing these values closer, it also brings closer the scores of images such as the ones in Figure 10, that get much more similar scores than in the method before, because all subpatches S_i are completely homogeneous. This means that we will get values close to zero for all distances d_{ij} in both cases, and it will all depend on the D_i values. In the left image, we have a quite high D_1 and much lower D_i values for $i = 2, 3, 4$, since the subpatches are more similar to the parent patch. In the right image, all of the 4 children subpatches have the same distance to the parent patch, getting a value similar to the average of all distances D_i in the left case.

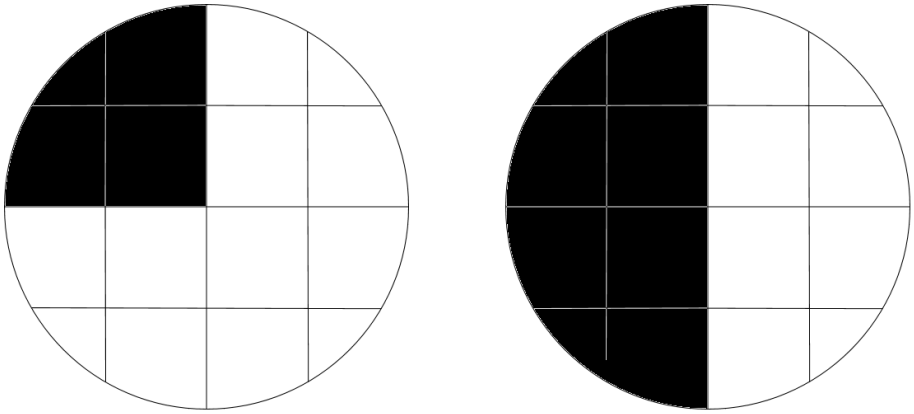

Figure 10: Disks with more similar heterogeneity scores in method 2 than in method 1

5.4.3 Method 3

Going back to the first method, one could wonder if averaging all the results would be a good idea or if we should penalize even more the existence of subpatches with a texture or color that's significantly different from the ones that define the whole patch.

If, instead of considering the average of all distances as the patch's heterogeneity score, we considered their maximum value, we would automatically label a patch as heterogeneous when it presents a significant irregularity, even if this is not too big, which may be useful for discarding disks that are not fully contained in a shape. Therefore, the algorithm for computing the cost in this case is as follows.

Let s_{1i} with $i \in 1, 2, ..., 4$ be the histograms that define each of the 4 coarser subpatches, and s_{2j}, where $j \in 1, ..., 16$ be the ones from the 16 finer parts. Also, let p be the histograms obtained from the disk. We define

$$D_{1i} = dist(p, s_{1i})$$

as the distance between the histograms in the subpatch s_{1i} and the patch p, and

$$D_{2j} = dist(p, s_{2j})$$

defines the difference between the part s_{2j} and the disk. The heterogeneity score, then, will be computed as

$$H = \max\left(\max_{i=1}^{4} D_{1i}, \max_{j=1}^{16} D_{2j}\right)$$

This method, however, can lead to some counterintuitive results, like the ones illustrated in Figure 11. While one would say the second disk is more homogeneous than the first one, since it has more pixels sharing the same value, it's not what this method computes. It compares each subpatch to the whole disk and finds a completely black subpatch in both cases, but the right parent patch has less black pixels in total than the left one, leading to a larger distance in histograms, and assigning a higher heterogeneity score to the right patch. This, however, could maybe be easily fixed if we divided each of the D_{1i} and D_{2j} distances by the area of the subpatch it's comparing to the parent patch, and then take the maximum of those weighted distances.

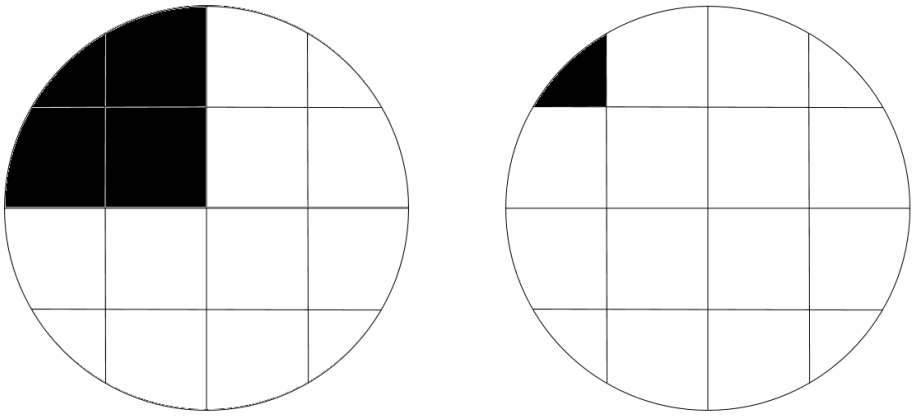

Figure 11: Patches with counterintuitive heterogeneity score assignments by method 3

5.4.4 Method 4

Following the same line of thought used in the method before, we can apply the same reasoning to the second of the methods we have seen. However, as the

second method is more complex than the first one, three variants can be created from it, shown in methods 4, 5 and 6.

In this first approach, we compute the heterogeneity score the same way we did in the second method, but taking the maximum of the final values that describe the quadrants of the disk, instead of its average. Therefore, defining S_i and s_{ij} as shown in Figure 9, we follow the same steps as in the second method until defining q_i for $i \in 1, ..., 4$ as we previously did. The difference between the two methods is right here, since the final heterogeneity score of the patch is now computed as

$$H = \max_{i=1}^{4} q_i$$

By applying this change, the scores assigned to disks such as the ones presented in Figure 10 get much more distant values that they did before, since now we are just considering the most different quadrant. This also means, however, that the method is less able to achieve rotational invariance than the second method.

5.4.5 Method 5

Another algorithm for computing the heterogeneity score of a patch can be derived from the second method. This option defines all patches and subpatches the same way it was done in the second one (Figure 9). It also compares all patches and subpatches as was done in that method, obtaining the same distances d_{ij} and D_i as we had, and even computing the heterogeneity h_i inside S_i the same way it was done there. The difference is that this method doesn't describe each quadrant as the average between its distance to the patch and the heterogeneity inside it, but takes it as the maximum of these two values:

$$q_i = \max\left(h_i, D_i\right)$$

And then takes the maximum of the q_i values as the heterogeneity score of the patch.

$$H = \max_{i=1}^{4} q_i$$

We now expect the results to be quite similar to the ones obtained when applying the forth method, since we are only considering the values in the most different quadrant in both methods. In this case, however, the small irregularities such as the ones in the right image in Figure 11 should give a higher score to the patch than they did in methods 2 or 4, although not as high as the one in method 3.

5.4.6 Method 6

Although more alternative methods could be developed from the second one, this is the last one we'll analyze, just to have an idea on how each change can affect the final results when applied to natural images. In this method, we won't just be changing the definition of q_i and H, but also the definition of the

h_i values, since, like the other ones, it was also defined as an average in method 2 and will be described as a maximum in this alternative method:

$$h_i = \max_{j=1}^{4} d_{ij}$$

From here, the algorithm goes on as in the previous method, defining q_i and H as:

$$q_i = \max\big(h_i, D_i\big)$$

$$H = \max_{i=1}^{4} q_i$$

As a result of this, the heterogeneity score of the patch is defined as the maximum of all distances D_i and d_{ij}, and can be written as:

$$H = \max\big(\max_{i=1}^{4} D_i, \max_{(i,j)|i,j=1}^{4} d_{ij}\big)$$

where the only difference against the third method is that the subpatches in the finer scale are now being compared to the ones in the coarser one, and not to the whole patch.

The expected results should give even more importance to the small irregularities in the image than the previous method was showing, obtaining for the right image in Figure 11 the same heterogeneity score we were getting when applying the third method, and the same score we would now get for the left image as well.

5.4.7 Method 7

All of the methods we have seen so far used in some way both children and grandchildren subpatches for computing heterogeneity scores, but as we have seen in the checkerboard example (Figure 5), the grandchildren division doesn't always provide better results, so it would make sense to also define some methods where we are only comparing the parent patch to its children subpatches. In this case, by applying the same idea we used in the first method, we are going to compare each of the four children patches in the disk to the parent patch.

Let s_i for $i \in 1, ..., 4$ describe each of the four subpatches in which the disk is divided. D_i defines the distance between s_i and the whole disk, p.

$$D_i = dist(s_i, p)$$

And the heterogeneity score of the patch is

$$H = \frac{1}{4}\big(\sum_{i=1}^{4} D_i\big)$$

As we've seen before, this method would obtain the expected score when applying it to a checkerboard image. The same thing happens in images such

as the one in the left image in Figure 12. However, it does not perform as well in images such as the right one in the same figure, where a high heterogeneity is expected (and obtained in all the previous methods), but the result is now approximately the same one that we have for the checkerboard, since histograms don't give information about geometry.

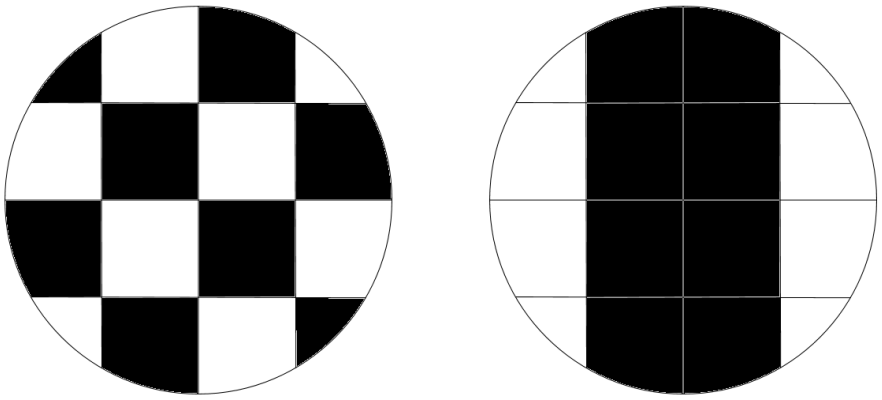

Figure 12: Patches with same heterogeneity score in method 7

5.4.8 Method 8

Following the trend from the previous methods, we should also consider computing the heterogeneity score of a patch as the maximum of the differences between its encodings and the ones defining each of its 4 children subpatches. Therefore, s_i, $i \in 1, ..., 4$ are the subpatches in which the patch p is divided. The heterogeneity score can be computed as:

$$H = \max_{i=1}^{4} dist(s_i, p)$$

And assigns the same scores as the method 7 to both images in Figure 12. When images have smaller irregularities, however, it assigns them a higher score, although not as high as the third method would obtain.

5.4.9 Method 9

Aiming for a different approach than the previous ones, this method is based on the idea that a homogeneous image is the one where all parts present similar features. Therefore, what we want is to compare each subpatch to all the other ones and average the results. For efficiency, we just use 4 subpatches to compute those differences. Let the different subpatches be described as s_i, with $i \in 1, ..., 4$. We compute the difference d_{ij} between all pairs s_i and s_j, where $i, j \in 1, ..., 4$, $i \neq j$

$$d_{ij} = dist(s_i, s_j)$$

and assign the heterogeneity score as the average of all those values:

$$H = \frac{1}{6}\Big(\sum_{(i,j)=1|i\neq j}^{4} d_{ij}\Big)$$

obtaining a very low score for the checkerboard and the other image in Figure 12, and a higher one for images with singularities, such as the ones in Figure 11, where the score is much higher than expected.

5.4.10 Method 10

Last but not least, it makes sense to compute heterogeneity score as the maximum of the differences between all of its subpatches. For achieving this value, we consider 4 subpatches s_i, with $i \in 1, ..., 4$ and compare each of them to all the other ones, computing the same distances $d_{ij} = dist(s_i, s_j)$ as in the previous method. The final score, however, will be the maximum of these distances instead of their average:

$$H = \max_{(i,j)=1|i\neq j}^{4} d_{ij}$$

The results we expect from the application of this method to different images are similar to the ones we had in the case before, except for some exceptions. Now, the scores assigned to all images where the two most different subpatches are the exact same ones, independently of what's in the other two subpatches, will have the same value. One example of this situation could be the two disks in Figure 10.

6 Testing the methods

For testing the methods on the whole set of images, we use the same function than in the AMAT method, that compares our results to the groundtruth medial axis points. For doing that, we first need to load all images contained in the dataset where we are performing the evaluation. The dataset we are using is the $BMAX500$, which is divided into three different sets of images: test, train and validation. In this case, just like in the AMAT paper, we are going to use the validation set for evaluation. Finally, we just evaluate our model on all images, create some dataset-wide statistics and store the results.

6.1 Preprocessing

Since we are comparing the medial axes obtained by our method to the one in the groundtruth, we need both items to be in the same format. When looking at the groundtruth medial axes offered by the dataset, it's easy to see the medial axes are represented by thin, defined lines (Figure 13C). Our results, however, are not as precise (Figure 13B). The pixels that comprise our medial axes are the ones where our previous algorithm decided a certain disk should be centered

for covering the whole image. Since the heterogeneity score of a pixel is usually similar to the ones in its neighbourhood, all pixels near the real axis should have very low scores. Because of this, and because we are just considering disks up to a certain scale, it's normal that in most cases the axes our method obtains, regardless of the method we are using for calculating the scores, is not as precise as the one in the groundtruth.

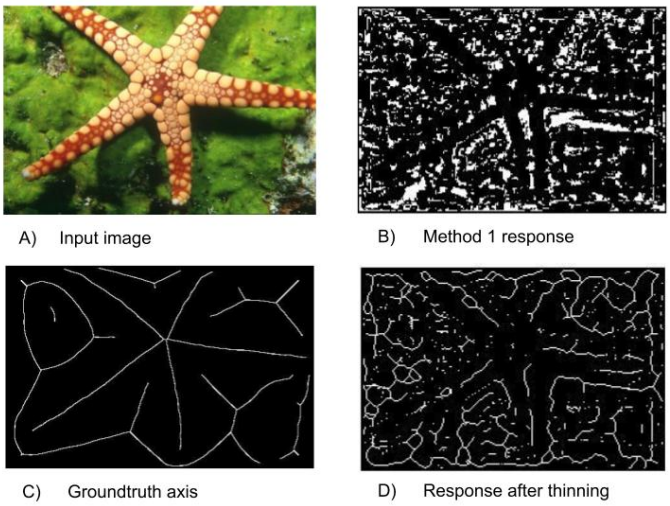

Figure 13: A) Input image from dataset BSDS500. B) Axis response obtained when applying our first method to the input image. Lines are thick and unclear. C) Medial axis points obtained from one of the ground truths offered from the dataset. D) Response from our method, shown in B), after thinning preprocessing is applied to it. Lines in here are thin and comparable to the ground truth

For solving that and bringing both axes representations closer, what we do, as also done in the AMAT, is apply an algorithm that groups the scattered points in our axes into branches, reduces the width of the resulting parts, and obtains thinner lines that can be actually compared to the groundtruth ones (Figure 13D). The way this algorithm performs is the following. In the first place, for each of the scales we have considered, we identify all the connected components in the MAT formed by points assigned to that scale. Then, we go through all connected components in each scale, from finer to coarser one, and consider a rectangle mask around each of them, while assigning them a new label.

We check the connected components in the immediate finer scales, which have already been labeled, trying to find another component that can be connected to the one we are studying. We consider two components are connected if their corresponding scales do not differ from more than 4 units and if they are close enough so that some of the pixels in one of the two components are in

the rectangular mask of the other one, to assure closeness and similarity. If our search is successful, we merge both components by assigning them the same label, which corresponds to the smallest one of all the different components being merged.

Once we have gone through all scales and every part of the axis has been assigned with a label, we build a label map and proceed to thinning each component into definite and precise curves. For doing that, we consider all pixels with the same label, called branches, one label at a time. For each label, we dilate all its points until they start merging with each other, to make sure all points conforming the branch are connected. Then, by using a built-in Matlab function, we can thin these points into a single line. When performing this on all the branches and joining all of the resulting branches together, we get final medial axes that are comparable to those in the ground truth of the dataset, and we can proceed to test our model.

It should be noted that our resulting branches can differ from the ones obtained in the AMAT, since we are grouping connected components by checking their proximity and similar scales, while the AMAT also checks color similarity between the two components, leading to possible differences in labelling. This, however, should not alter the final medial axes, but could make a difference if we are focusing on the different branches in which it is divided.

6.2 Evaluation of the model

For evaluating the model, once all images in dataset have been loaded with their corresponding groundtruth axes, and the skeleton obtained with our approach has been preprocessed, we compare our result to the ones in the groundtruth for every image and compute some statistics.

The diagnosis for each pixel when comparing the two medial axis distributions can be divided into four different groups:

- True positive (TP): The pixel appears to be in the axis in both our result and the groundtruth.
- True negative (TN): The pixel doesn't appear to be in the axis in either our result nor the groundtruth.
- False positive (FP): The pixel appears to be in the axis in our result but not in the groundtruth.
- False negative (FN): The pixel appears to be in the axis in the groundtruth but not in our result.

By dividing the true positives by the sum of true and false positives, we get the precision P of the method on the image, while, if we divide the true positives by the sum of false negatives and true positives, we get the recall R.

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

If we call relevant pixels to the ones where the groundtruth places a medial point, we can understand the precision as the proportion of relevant pixels in the axis obtained by our approach, giving us an idea of the exactness of the method, while the recall (or sensitivity) gives the proportion of relevant pixels that are selected by our method, providing a taste of its completeness.

By combining these two results, we can measure the test accuracy computing the $F1 - score$:

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

where P is the precision and R the recall and the final score F corresponds to the harmonic average between both numbers. The closer the F-score gets to 1, the more accurate the method seems to be, while if it is close to 0, the method doesn't appear as a very good one. By linear interpolation of all the F-points found in the images, the best one is computed and shown in the final plot, as the representative F-score of the method.

7 Results

When testing the AMAT on the previously smoothed images in the validation set of $BMAX500$, by using the previous testing algorithm, we obtain an F-score of $F = 0.5679$, with a precision of $P = 0.5067$ and a recall of $R = 0.6458$. If the images weren't smoothed, however, the evaluation would have an F-score of just $F = 0.5466$, with a precision of $P = 0.4646$ and a recall of $R = 0.6636$. Obviously, the method performs better on the smoothed images, since that's actually the reason for smoothing them, but what may come as a surprise is that the recall value of this method is higher when the images are not being smoothed, meaning that we obtain less false negatives in this case. This actually makes sense, since the algorithm building the medial axis wants to place disks where the reconstruction error is minimal, and so, where the average color does not diverge too much from all the colors inside it. When the images are textured and there's no smoothing applied, the algorithm is going to choose smaller disks to cover the image by reducing the reconstruction error, taking a larger number of disks and, therefore, a lower amount of both false negatives and true negatives.

Let's now study the results our different methods get when performing on the same set of images.

Method	Precision	Recall	F-Score
AMAT with Smoothing	0.5067	0.6458	0.5679
AMAT without Smoothing	0.4646	0.6636	0.5466
Method 1	0.4856	0.5273	0.5056
Method 2	0.4997	0.4832	0.4913
Method 3	0.4746	0.5222	0.4972
Method 4	0.4994	0.4709	0.4847
Method 5	0.4901	0.5067	0.4983
Method 6	0.4769	0.5204	0.4977
Method 7	0.4994	0.3787	0.4307
Method 8	0.4812	0.3315	0.3926
Method 9	0.4963	0.3978	0.4416
Method 10	0.4895	0.3800	0.4279

Table 2: Results obtained applying AMAT and all of our studied methods on the validation set of images in BSDS500.

Looking at the results in the table above (Table 2), we see all our methods get a lower F-score than the AMAT does, but that doesn't mean they always perform worse, it's just an overall view on the results. For example, we can see how all methods apply to one same image (Figure 14), with a performance that speaks better for some of our methods than it does for the AMAT (Table 3).

Method	Precision	Recall	F-Score
AMAT with smoothing	0.4503	0.6406	0.5288
AMAT without smoothing	0.4138	0.6078	0.4924
Method 1	0.4967	0.5661	0.5291
Method 2	0.5394	0.5567	0.5479
Method 3	0.4914	0.5397	0.5144
Method 4	0.5619	0.5240	0.5423
Method 5	0.5166	0.5377	0.5270
Method 6	0.4940	0.5392	0.5156
Method 7	0.6451	0.3802	0.4784
Method 8	0.6626	0.3365	0.4464
Method 9	0.6258	0.3860	0.4775
Method 10	0.6005	0.3656	0.4545

Table 3: Evaluation for all the methods applied to the image in Figure 14

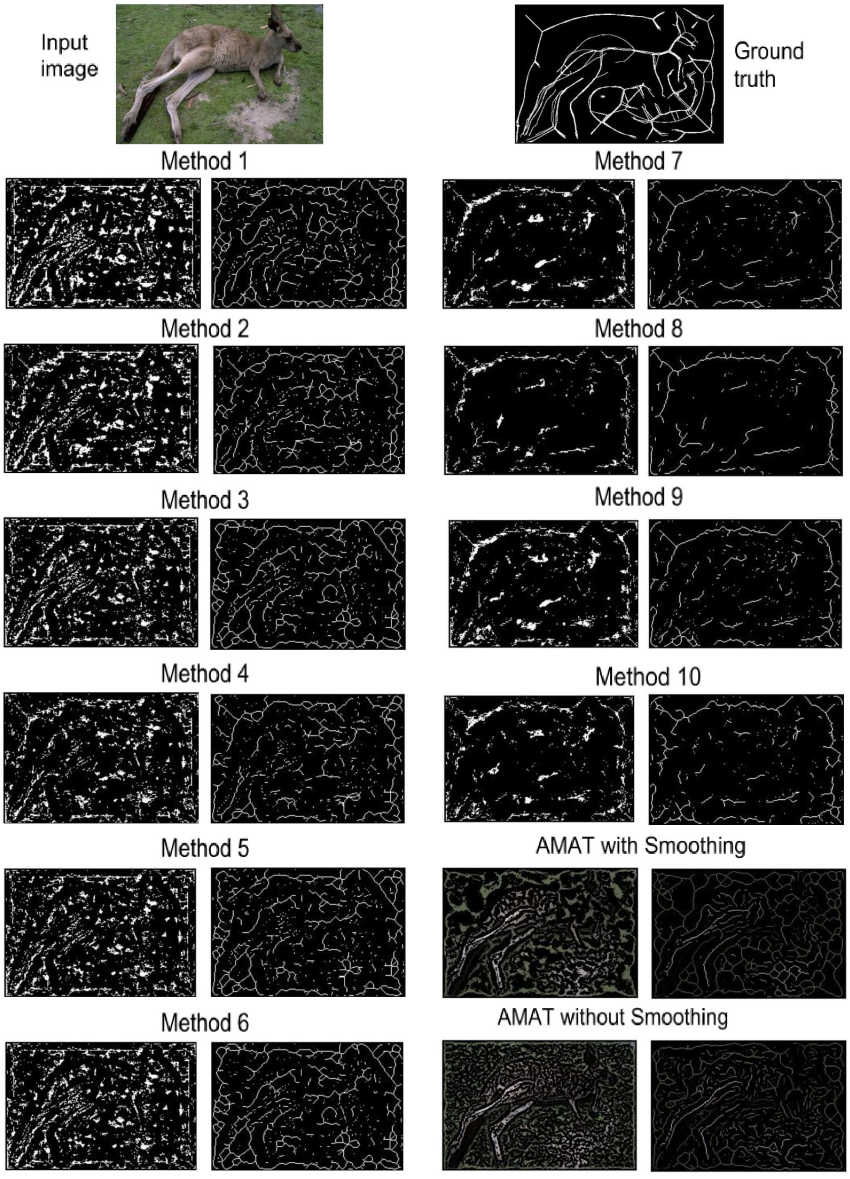

Figure 14: Response for all methods to an input image from the validation set in BSDS500. Shown: Input image, union of all the ground truths to which the methods are compared for evaluation, results from each method from 1 to 10, and AMAT with and without smoothing, all before and after thinning the skeleton.

Looking at the performance of every method on this image (Figure 14, Table 3), we see that three of our methods perform better than the AMAT itself, and, if compared to the AMAT without previous smoothing, for having the same starting point than in our methods and, therefore, a fairer comparison, we get a better result for more than half of the methods we are presenting. We see, with this example, that the F-score doesn't give us a clear and absolute truth about the performance of the different methods on all kinds of pictures, but gives us a general idea on how they apply on a certain set of images. However, for a deeper interpretation on the results, let's now focus on the precision and recall values we are obtaining for each method.

When looking at the recall value, it's easy to see we are getting a high variation between the methods, and in all cases, the results are notably worse than in the AMAT, both in the example we have studied (Table 3) and in the results for the whole dataset (Table 2). Getting a low recall value means the number of false negatives is high, and therefore, comparing to the groundtruth, it means our result is missing many medial points.

Among all obtained values for R, the methods that get the lowest ones are the last 4, the ones where only four subpatches are being considered in the disks for computing heterogeneity scores. The fact that they are missing many medial points could indicate they are assigning the medial category to the wrong points, which would lead to an equally high number of false positives. However, their precision is not equally low, so the number of false positives is quite lower than the one of false negatives. This can just mean that our methods' results have less medial points than the ground truths. Since every medial point corresponds to a disk, and all points in the image should be covered by at least one of those disks, obtaining less medial points than we should can only mean we are covering the image by bigger disks, driving us to wrong results and less medial points than expected.

Let's view this reasoning with an example. Considering the image of the sea star shown in Figure 13A, and comparing in Figure 15 the results we are getting when using both the first and the eighth method (since they are, respectively, the ones that obtain the highest and lowest recall value on the validation set), we can observe the described behaviour.

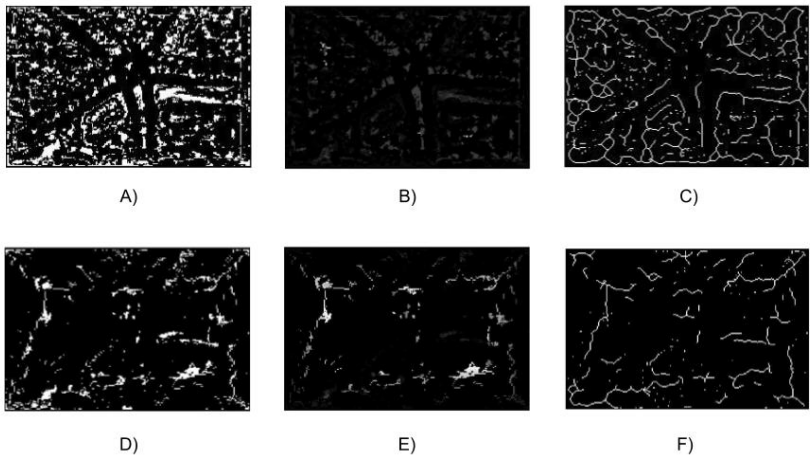

Figure 15: Comparison of results obtained from the sea star image in Figure 13A. First row: Result obtained applying the first method to the image. Second row: Results from applying the eighth method. First column: Medial points detected by the methods, without grouping or thinning them. Second column: Scale assigned to every medial point. Black means 0 and white is the maximum considered scale. Third column: axis obtained when grouping and thinning the medial points.

Clearly, most of the medial points from the first method, shown in the first row, are associated to much finer scales than the ones from the eighth method, leading to an axis much more similar to the groundtruth in the first case.

The reason behind this relies on the much lower heterogeneity score the eighth method assigns to some kind of patches, than the first method. In the methods that consider only 4 subpatches, such as the eighth one, obtaining a low heterogeneity score is much easier than in the methods that divide the patch into 16 parts, like the first method. When the subpatches we are comparing are very large, some smaller irregularities inside of it may go by unnoticed, without altering much the histograms. This results in a lower distance between the subpatch and other similar subpatches or the parent patch, and as a consequence, a lower heterogeneity score. If we considered smaller subpatches, however, the probability of capturing the irregularity of the image in a subpatch, having a huge impact on the histograms, is much higher, leading to larger histogram distances and higher scores.

However, if comparing the axis obtained for this last image when using each of these methods to the ones from the AMAT and the groundtruth (Figure 16, Table 4), we can easily realize the first method is the one that gets a closer response to the ground truth for the star, obtaining also a higher F-score (Table 4), but it's also the one that performs the worst on the background. The reason behind this is the exact same one that also explains the good results for the

foreground. Since the background is formed by quite different shades of green and shows also some shadows and irregularities, when placing big disks in it and comparing them with its smaller subpatches, they will be assigned with a quite high heterogeneity score. For this reason, smaller disks are used for covering the background, and therefore, a large amount of false positives and false negatives are obtained. When using the eighth method, on the other hand, as we have seen, heterogeneity scores don't grow as much with this kind of regularities, obtaining for the background a much closer skeleton to the one in the ground truth.

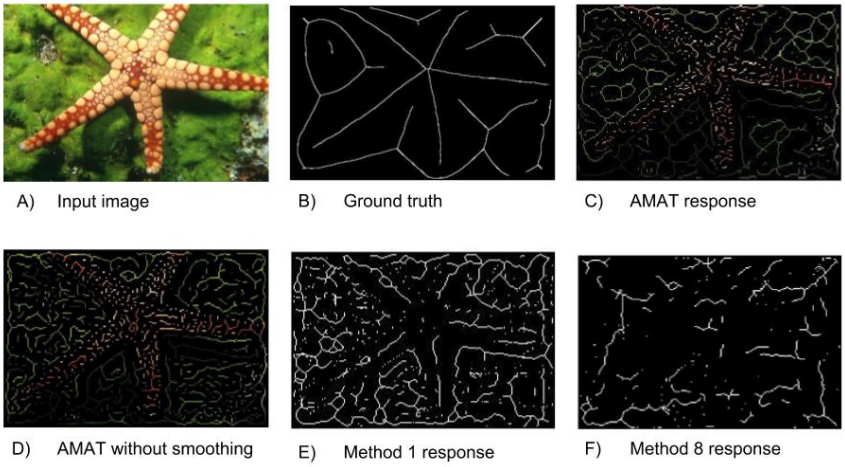

Figure 16: Comparison of skeleton obtained from the input image. A) Input image. B) Ground truth provided by the dataset. C) Skeleton obtained applying AMAT with previous smoothing of the image. D) Skeleton from AMAT without smoothing. E) Medial axis points from method 1. F) Medial axes from method 8.

Method	**Precision**	**Recall**	**F-Score**
AMAT with smoothing	0.3771	0.6437	0.4756
AMAT without smoothing	0.3776	0.6858	0.4870
Method 1	0.4658	0.6008	0.5248
Method 8	0.5058	0.2981	0.3751

Table 4: Evaluation on the image in 16 for some significant methods.

This weak side of the first method is very well exemplified from studying the properties of the image where it gets the lowest F-score, shown in Figure 17.

Figure 17: Image from validation set in dataset BSDS500 with the worst performance from the first method.

This is a very challenging image, since, to get good results, the method should be able to detect the skeleton for both very broad regions and very small ones, like the animal in the picture. However, we are just considering a very specific set of scales for the disks, that go from 2 to 41 pixels radii. For this reason, we may not be considering the maximal disks in the background, resulting to quite bad results. However, for obtaining the closest result to the groundtruth, we should be taking as medial points the centers of the biggest available disks. As we have seen, since the background is not too homogeneous, the methods that will perform best in this image will be the ones where only four subpatches are being considered, where small irregularities don't make heterogeneity scores grow too much.

Method	Precision	Recall	F-Score
AMAT	0.2486	0.5750	0.3471
Method 1	0.1485	0.5142	0.2305
Method 2	0.1724	0.5405	0.2614
Method 3	0.1534	0.5049	0.2353
Method 4	0.1829	0.5708	0.2770
Method 5	0.1759	0.5893	0.2710
Method 6	0.1569	0.5155	0.2406
Method 7	0.1907	0.5166	0.2785
Method 8	0.1941	0.5000	0.2796
Method 9	0.1849	0.5511	0.2768
Method 10	0.1833	0.5005	0.2683

Table 5: Statistics for the image in Figure 17

In the table 5, we can see how, in the case of this last image, the method that gets the best results is the eighth one, while the first one gets the most distant results from the groundtruth. So, observing different images, we can appreciate how the methods that consider only 4 subpatches of the disks for computing the heterogeneity scores obtain better results for images with coarse parts to detect, while, when the image has finer parts that should be represented in the final skeleton, the methods that consider more subpatches get way better results.

Now that the huge differences between the methods' recall values have been explained, we should focus on the precision. When we are getting a low P value, it means the number of false positives is quite high, so we are assigning a lot of medial points in places where the groundtruth doesn't. As we can see, in absolutely all of our methods we are getting very similar results for the precision, slightly lower than the one for the AMAT with smoothing and slightly higher than the one without smoothing.

In general (Table 2), the method that performs better in this sense is the second one, while the one that obtains more results far from the ground truth, is the third one. When a method gets a high number of false positives, it means it's considering different medial points than the ground truth. The reason that happens is that there's a problem in the distribution of heterogeneity scores and the method is assigning a lower score to disks that do not correspond to medial points than to the other ones, favoring the selection of the wrong points. This being said, we can see that, overall, the method that gets the most wrong heterogeneity score distribution is the third one, because small irregularities inside a disk can lead to huge heterogeneity scores, as we have already seen, leading the algorithm to choose other disks where the heterogeneity score is a little lower, even if they are actually worse medial disks candidates.

It's this exact mistake that makes the third method be worse than the first one, the eighth method be worse than the seventh one and the tenth method perform worse than the ninth one, so we see that taking the maximum of the calculated differences performs worse, overall, than taking their average and considering the differences in the whole disk.

For the second method and its three different variations (methods 4, 5 and 6), another reasoning is applied, since taking a maximum or an average changes the general structure of the method more than it did in the other cases. In the forth case, we are using the exact same algorithm than in the second method, but considering just the most different quadrant, instead of the average for all the four ones, getting the same effect we had for the previous cases, and therefore, obtaining more false positives and false negatives and as a result, a lower F-score and a worst performance.

For the fifth and sixth methods, we are getting less false negatives, but slightly more false positives than in the second one, meaning that, in total, we are obtaining more medial points than in the previous case. When this happens, as we have seen, it means that smaller disks have been used for covering the whole image, so higher scores must have been assigned to the largest disks. This makes sense, since we were before taking the average between the results from the smaller subpatches and the biggest ones, and we are now taking the maximum between those two, and, as expected, these values are going to be higher.

8 Conclusions and Future Work

Taking everything into consideration, it's safe to say we have presented a decent approach to obtaining the medial axes of natural images without previous smoothing nor segmentation. After providing different methods and studying how they all apply to a set of images, the one that shows a better performance overall is the first of them all. However, it should be noted that its weakest point is the strongest point of a different kind of method, such as the seventh one. Therefore, in the future, a mixture of them could be computed, getting the best of the both algorithms and performing even better on the set of images.

The key for obtaining better results, then, could be in developing an intelligent method that compares different number of subpatches depending on the image we are considering, or maybe one that, despite comparing many small subpatches, can relax a little the comparison between all of them. And this is actually what we had tried in methods 2, 4, 5 and 6, in higher or lower degree, obtaining, as we can see, performances that are in between of the other methods. However, in those methods, the grandchildren subpatches are being compared to the children ones, but it could also make sense comparing them to the whole patch, just as we did for the first method, but weighting differently the distances between the patch and each of the two subpatches groups. Therefore, an approach worth trying would be, as we have commented in some of the methods, altering them a little by weighting the distances involving children and grandchildren subpatches proportionally to their areas, assigning a larger weight to the distances between the parent patch and the coarser subpatches and therefore, assigning higher heterogeneity scores to disks with big irregularities than to disks with smaller ones.

Something else to think about would be not only considering different ways

of combining the histogram distances obtained for each subpatch, but also trying different combinations of histogram distances for every channel. Maybe if texture and luminance had more weight than color, the irregularities shown in the background wouldn't alter the scores that much and their axes would be smoother and more similar to the ground truth, yielding a better evaluation.

In conclusion, it may not perform as well as the AMAT does on average, but it does outperform it on certain kinds of images, where there's a lot of texture, but it's substantially regular throughout each element in the image, and especially if all those elements are of a similar scale. When the textures have a lot of irregularities, however, our algorithm doesn't perform as well, and certain smoothing could still come in handy.

References

[1] Sven Dickinson. *The Evolution of Object Categorization and the Challenge of Image Abstraction*.

[2] Sven J. Dickinson and Zygmunt Pizlo. *Shape Perception in Human and Computer Vision: An Interdisciplinary Perspective*. Springer Publishing Company, Incorporated, 2013.

[3] D. H. Ballard and C. M. Brown. *Computer Vision*. New Jersey, USA: Prentice Hall, 1982. URL: http://homepages.inf.ed.ac.uk/rbf/BOOKS/BANDB/bandb.htm.

[4] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York, NY, USA: Henry Holt and Co., Inc., 1982. ISBN: 0716715678.

[5] Patricio Simari and Karan Singh. *Extraction and remeshing of ellipsoidal representations from mesh data*. Jan. 2005.

[6] David Therriault. *On the Future of Object Recognition: The Contribution of Color*. Apr. 2011.

[7] Tijan Mede et al. "A medial axis based method for irregular grain shape representation in DEM simulations". In: *Granular Matter* 20.1 (Jan. 2018), p. 16. ISSN: 1434-7636. DOI: 10.1007/s10035-017-0785-7. URL: https://doi.org/10.1007/s10035-017-0785-7.

[8] Thomas O. Binford. "Visual Perception by Computer". In: *Proceedings of the IEEE Conference on Systems and Control (Miami, FL)*. 1971.

[9] Steven Shafer and Takeo Kanade. "The Theory of Straight Homogeneous Generalized Cylinders and A Taxonomy of Generalized Cylinders". In: *Proceedings of the 1983 DARPA Image Understanding Workshop*. Jan. 1983, pp. 210–218.

[10] Barr. "Superquadrics and Angle-Preserving Transformations". In: *IEEE Computer Graphics and Applications* 1 (1981), pp. 11–23.

[11] Irving Biederman. “Recognition-by-components: A theory of human image understanding”. In: *Psychological Review* 94 (1987), pp. 115–147.

[12] K. Koffka. *Principles of Gestalt Psychology*. New York: Harcourt, Brace, 1935.

[13] Harry Blum. “A Transformation for Extracting New Descriptors of Shape”. In: *Models for the Perception of Speech and Visual Form*. Ed. by Weiant Wathen-Dunn. Cambridge: MIT Press, 1967, pp. 362–380.

[14] Wei Shen et al. “Skeleton pruning as trade-off between skeleton simplicity and reconstruction error”. In: *Science China Information Sciences* 56.4 (Jan. 2013), pp. 1–14. DOI: 10.1007/s11432-012-4715-3. URL: https://doi.org/10.1007%2Fs11432-012-4715-3.

[15] Roger Tam and Wolfgang Heidrich. “Shape Simplification Based on the Medial Axis Transform”. In: *IEEE Visualization*. Oct. 2003, pp. 481–488.

[16] Y.-C. Chang et al. “Medial Axis Transform (MAT) of General 2D Shapes and 3D Polyhedra for Engineering Applications”. In: *Geometric Modelling*. Springer US, 2001, pp. 37–52. DOI: 10.1007/978-0-387-35490-3_3. URL: https://doi.org/10.1007%2F978-0-387-35490-3_3.

[17] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. “Free-form Skeleton-driven Mesh Deformations”. In: *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*. SM ’03. Seattle, Washington, USA: ACM, 2003, pp. 247–253. ISBN: 1-58113-706-0. DOI: 10.1145/781606.781643. URL: http://doi.acm.org/10.1145/781606.781643.

[18] Wei Shen et al. “Object Skeleton Extraction in Natural Images by Fusing Scale-associated Deep Side Outputs”. In: *CoRR* abs/1603.09446 (2016).

[19] Stavros Tsogkas and Iasonas Kokkinos. “Learning-Based Symmetry Detection in Natural Images”. In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 41–54. ISBN: 978-3-642-33786-4.

[20] Kaleem Siddiqi et al. “Shock Graphs and Shape Matching”. In: *International Journal of Computer Vision* 35.1 (1999), pp. 13–32.

[21] Diego Macrini et al. “Bone Graphs: Medial Shape Parsing and Abstraction”. In: *Comput. Vis. Image Underst.* 115.7 (July 2011), pp. 1044–1061. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2010.12.011. URL: http://dx.doi.org/10.1016/j.cviu.2010.12.011.

[22] T. Lindeberg. “Edge detection and ridge detection with automatic scale selection”. In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. June 1996, pp. 465–470. DOI: 10.1109/CVPR.1996.517113.

[23] S. Dickinson A. Levinstein C. Sminchisescu. “Multiscale symmetric part detection and grouping”. In: (). DOI: 10.1109/ICCV.2009.5459472.

[24] Stavros Tsogkas and Sven J. Dickinson. "AMAT: Medial Axis Transform for Natural Images". In: *CoRR* abs/1703.08628 (2017). arXiv: 1703.08628. URL: http://arxiv.org/abs/1703.08628.

[25] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall Information and System Sciences Series. Prentice-Hall, 1989.

[26] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.5 (May 2004), pp. 530–549. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.1273918. URL: http://dx.doi.org/10.1109/TPAMI.2004.1273918.